REMARKS

This is intended as a full and complete response to the Office Action dated November 2, 2005, having a shortened statutory period for response set to expire on February 2, 2006. Please reconsider the claims pending in the application for reasons discussed below.

Claims 1-26 are pending in the application. Claims 1-26 remain pending following entry of this response. Claims 1, 12 and 16 have been amended. Applicants submit that the amendments do not introduce new matter.

Claim Rejections - 35 U.S.C. § 103

Claims 1-26 are rejected under 35 U.S.C. § 103(a) as being unpatentable over US. Patent No. 6,263,489 to Olsen et al. (*Olsen*) in view of U.S. Patent No. 6,161,216 to *Shagam* and further in view of "An Efficient Relevant Slicing Method for Debugging" by Gyimothy et al. (*Gyimothy*).

Applicants respectfully traverse this rejection.

The Examiner bears the initial burden of establishing a *prima facie* case of obviousness. *See* MPEP § 2142. To establish a *prima facie* case of obviousness three basic criteria must be met. First, there must be some suggestion or motivation, either in the references themselves or in the knowledge generally available to one ordinary skill in the art, to modify the reference or to combine the reference teachings. Second, there must be a reasonable expectation of success. Third, the prior art reference (or references when combined) must teach or suggest all the claim limitations. *See* MPEP § 2143. The present rejection fails to establish at least the first and third criteria.

For example, *Olsen*, in view of *Shagam* and *Gyimothy*, fails to disclose a method for a debugger application to select locations for inserting breakpoints in a program being debugged that includes the step of selecting branch points at which to insert the breakpoints, as recited by claims 1, 12, and 16.

The Examiner Asserts that "Olsen discloses a computer-implemented method for inserting breakpoints in a program being debugged (see, for example, the abstract, and step 54 in FIG. 4A, which shows inserting breakpoints)" and that "Olsen discloses selecting instructions in the program at which to insert breakpoints (see, for example, column 13, lines 8-12)." See *Office Action*, p.4. In fact, the cited passages are directed to actions performed <u>only after a user has selected a location at which to insert a breakpoint</u>. The material cited by the Examiner is one of several examples described in *Olsen* wherein a <u>user</u> selects a breakpoint location, including:

> "The user sets a breakpoint at a breakpoint P in the source code where execution is to stop." *Olsen*, 3:2-5.
> "Assume the user sets a breakpoint at source line 2." *Olsen*, 4:30.
> "When the user sets a breakpoint at source location P... *Olsen*, 4:64.
> "Thus., when the user sets a source breakpoint at P" *Olsen* 5:27-30.
> "Initially, the user sets a breakpoint P in the application source." *Olsen*, 12:1-2.
> "the user enters a command "set breakpoint at P (step 50)." *Olsen*, 12:18-19, Fig. 4A, step 50.

In response to the <u>user-specified</u> selection of a location for a breakpoint, *Olsen* discloses a technique for inserting the breakpoint in optimized code. However, there is nothing in *Olsen* regarding a method for a debugger application to select locations for inserting breakpoints in a program being debugged.

Further, *Olsen* does not teach selecting branch points at which to insert the breakpoints, whether by a user operation or programmatically by a debugger application. Therefore, the rejection is believed to be improper and Applicants respectfully request that the rejection be withdrawn.

In addition to these fundamental distinctions between *Olsen* and the present claims, the Examiner goes on an asserted equivalence between *Shagam*'s use of "trace points" as the equivalent of a "break point." See Office Action, p.4. These two data structures, however, are neither the same nor even analogous. As described in *Shagam*:

> Each trace point location in the specification includes a pass-count for

> determining the number of times the trace point location is executed before terminating execution of a process associated with the source code.
>
> ...
>
> The trace points are executable instructions for collecting stored information (e.g., in registers) and indicate to the programmer, a flow of the source code, and whether the source code is being executed in the manner anticipated and intended by the programmer. Source code debugger 16 generates to a target machine 28, a debugger trace component 17 which is part of executable code 19.

Shagam, 1:64-66, and 3:43-48. At the same time, Shagam recognizes the difference between a "Trace point" and break point as follows:

> Execution of the source code program is halted whenever a break point is encountered to allow the programmer to observe certain variables and the behavior of the program at the breakpoint.

Shagam, 1:22-25. *Shagam* discloses techniques for a debugger "script generator" to generate "debugging scripts" performed at each "trace point" location. Applicants submit that "trace points" and "break points" however, are neither equivalent, nor analogous, as they are used to perform fundamentally different actions. Namely, a "trace point" executes data gathering instructions and "pass counting" functions whenever encountered, whereas break point is used to halt execution of a running program and turn over execution control to a debugger application. Again, this is a distinction that *Shagam* itself recognizes.

Lastly, recognizing that *Olsen*, in view of *Shagam* fails teach or suggest all of the limitations of the present claims, the Examiner turns to *Gyimothy*. Specifically, the Examiner concedes that *Olsen* in view of *Shagam* does not "expressly disclose the recited limitation of determining which other blocks present in the program control execution of the basic block." See *Office Action*, p.5.

Gyimothy, however, provides a debugging system that relies on a structure referred to as a "program dependence graph." As defined in *Gyimothy* the "program dependence graph contains "one vertex for each statement and control predicate." *Gyimothy*, p.2, left column last paragraph. That is, one graph node for each single instruction. Further, the "program dependence graph" includes two types of edges between nodes "data dependence edges and control dependence edge." *Gyimothy*, p.2, right column, first paragraph. As should be clear from *Gyimothy*'s definition, the "program dependence graph" is a graph where each node represents an individual statement and wherein the edges reflect different relationships among program statements. In contrast, a "block" is a sequence of consecutive statements in which flow of control enters at the beginning and leaves at the end without halt or the possibility of branching except at the end of the block. Thus, each "block" may include multiple program statements. Therefore, Applicants submit that the "program dependence graph" of *Gyimothy* fails to disclose anything regarding a block of execution at all, and is instead related to a graph of relationships among individual program statements.

Accordingly, for all the foregoing reasons, Applicants submit that claims independent claims 1, 11, and 16, and the claims dependent therefrom, are allowable and respectfully request, therefore, that the rejection be withdrawn and the claims be allowed.

<u>Conclusion</u>

If the Examiner believes any issues remain that prevent this application from going to issue, the Examiner is strongly encouraged to contact Gero McClellan, attorney of record, at (336) 643-3065, to discuss strategies for moving prosecution forward toward allowance.

Having addressed all issues set out in the office action, Applicants respectfully submit that the claims are in condition for allowance and respectfully request that the claims be allowed.

Respectfully submitted, and
S-signed pursuant to 37 CFR 1.4,

<u>/Gero G. McClellan, Reg. No. 44,227/</u>
Gero G. McClellan
Registration No. 44,227
PATTERSON & SHERIDAN, L.L.P.
3040 Post Oak Blvd. Suite 1500
Houston, TX 77056
Telephone: (713) 623-4844
Facsimile: (713) 623-4846
Attorney for Applicants